Relational Regularization and Feature Ranking

Fabrizio Costa* Mathias Verbeke† Luc De Raedt†

Abstract

Regularization is one of the key concepts in machine learning, but so far it has received only little attention in the logical and relational learning setting. Here we propose a regularization and feature selection technique for such setting, in which one commonly represents the structure of the domain using an entity-relationship model. To this end, we introduce a notion of locality that ties together features according to their proximity in a transformed representation of the relational learning problem obtained via a procedure that we call "graphicalization". We present two techniques, a wrapper and an efficient embedded approach, to identify the most relevant sets of predicates which yields more readily interpretable results than selecting low-level propositionalized features. The proposed techniques are implemented in the kernel-based relational learner kLog, although the ideas presented here can also be adapted to other relational learning frameworks. We evaluate our approach on classification tasks in the natural language processing and bioinformatics domain.

1 Introduction

In inductive logic programming [1] and statistical relational learning [2], the input data is represented using expressive *logical and relational* representations [3]. The examples consist of entities and relations, rather than attribute-value or feature vector representations, which are essentially propositional. Each example is typically represented as an interpretation, a kind of small relational database (a possible world), listing the tuples that are true for that example. Furthermore, background knowledge can often be used to define predicates in a declarative way. The conciseness of relational representations and the use of background knowledge are among the key advantages of logical and relational learning [3].

The conciseness of relational representations is manifest when considering their corresponding *propositionalized* representations [4]. These are used to define the semantics of statistical relational models (in knowledge-based model construction approaches [2] such as Markov Logic [5]) and also as the basis for effective learning approaches [6, 4, 7]. In all these approaches a large number of propositional features is derived from the original relational representation. While there exist many techniques to regularize propositional models and to select propositional features in order to gain insight into the relative importance of these features, the problem of regularization and feature selection is not yet well understood for the *relational* representations.

A regularization procedure, in machine learning, is a way to introduce information, independently from the evidence present in the data, in order to solve an ill-posed problem or to prevent overfitting. Typically one tries to enforce simplicity constraints, for example reducing the effective degrees of freedom in a model by tying together the model parameters in some prescribed way. In this paper, we introduce a novel type of regularization, called *relational regularization*, that takes into account the relational structure of the domain as specified in an entity-relationship (E/R) model. These relations are used to tie the parameters of a predictive linear model, and we show how one can combine the importance score assigned to the different features to obtain the importance for the original elements in the E/R diagram.

Finally, based on these concepts, we present an embedded and a wrapper approach for *relational feature selection and ranking*, which allows us to get deeper insights into the relative importance of the elements in the E/R models used to describe respectively a natural language processing and a bioinformatics problem.

The proposed techniques are implemented in kLog [7], a relational learning system that is competitive (publication under review) with respect to other systems such as Aleph's [8] Markov Logic Networks [5] and Tilde's [9] first order logical decision trees. However, the techniques presented can also be adapted and ported to other relational learning frameworks.

Related Work Feature selection has a long tradition in machine learning and has also been studied for logical and relational learning. For instance, Alphonse and Matwin [10] upgrade propositional methods for feature selection in relational learning using a transformation to multi-instance learning rather than to graphs but they neither employ a notion of locality nor a notion of regularization. Jensen and Neville [11] analyze the feature selection bias caused by linkage and autocorrelation, two notions capturing important properties about the topology of relational data. They also use this to leverage the performance of relational learners [12], but do not provide a regularization approach. Regularization has been used for learning the structure of Markov Logic networks, e.g., Huynh and Mooney [13] first derive a large set of clauses and then determine their weights

*Institut für Informatik, Albert-Ludwigs-Universität, Germany, costa@informatik.uni-freiburg.de.

†Department of Computer Science, KU Leuven, Belgium, firstname.lastname@cs.kuleuven.be.

using L_1 regularization. As the set of clauses is fixed in the second step, they apply propositional regularization techniques which do not take into account the relational topology. Quanz and Huan [14] extend the L_2 regularized logistic regression to aligned graph classification, however a propositional (vector) representation is used. Furthermore, all instances have the same graph structure, where the relationships are assumed known and fixed. In our case, the graphs associated to examples are all different, and generated declaratively. Finally, Zhou and Schölkopf [15] propose a general regularization framework for graphs, however they do not work within a logical setting and they deal with single graph domains. In this work, instead, we do not consider graphs representing relations between instances, but rather use a graph representation to encode the relations between *parts* of instances defined in a logical framework.

2 Overview

kLog: a Logic-based E/R System kLog [7] is a logical and relational language for kernel-based learning, that is embedded in Prolog, and builds upon and links together concepts from database theory, logic programming and learning from interpretations. In essence one can use kLog to represent in a very concise and very high-level fashion both data, prior knowledge and the problem description. kLog employs a logical and relational data representation rooted in the E/R model. For each example, kLog then computes the corresponding *graphicalization*, that is, it declaratively converts instances into graphs. In practice, the graphicalization corresponds to the *unrolled* E/R model of the example, which lists all relations that hold in the example. With the proposed regularization, different related parts of the instances are tied together in a coherent way, using the graph structure. We thus use kLog as a system to simplify the design phase and perform regularization and feature selection on this meta-level.

Artificial Example We will first illustrate the concepts and techniques used in this paper by means of the following artificial game. Equipped with a bag filled with cubes and spheres, the game consists of drawing a random sequence of 10 objects from the bag. Each object has a certain color and weight. The game can only be won if two *spheres* with the same color are drawn without drawing two *cubes* with the same color. In any other case the game is lost.

The E/R diagram for this game is shown in Figure 1 (top left). One particular game (that is, *instance*) is shown in Figure 1 (bottom left), in a relational notation, and in Figure 1 (top right) as the unrolled E/R diagram for this case. In kLog's terminology, the unrolled E/R diagram is essentially the same as the *graphicalized instance*.

In logical and relational learning it is common to employ *background knowledge* to define particular relations. In the above example, it would be quite natural to have knowledge about the color and the weight of the objects, which could be represented as sphere(s1,r,2), cube(c1,g,1), etc. (i.e., a red sphere of 2 kg, and a green cube of 1 kg). In addition, predicates such as sph_eq_col could then be defined in Prolog as sph_eq_col(X,Y) :- sphere(X,C,_), sphere(Y,C,_). which defines spheres of equal color. The E/R diagram in Figure 1 shows the relevant relations for our example. For ease of exposition, we shall mostly use the E/R representation in the rest of this paper, though the reader should keep in mind that the kLog language allows the user to define this type of E/R diagram and background knowledge in Prolog, reminiscent of inductive logic programming systems.

Relational Regularization and Feature Selection Our proposal is based on two core ideas. The first is to define features on the E/R diagram, called *high-level (relational) features*, and features on the grounded version of the E/R model of each example, called *low-level (graph) features*. The second idea is to tie together sets of low-level features into high-level features.

From a relational perspective, the features correspond to queries. The *low-level features* are derived from the *unrolled* E/R diagram and correspond to (particular) subgraphs of a graphicalized instance[1]. Some possible low-level features are illustrated in Figure 2, e.g., one possible feature corresponds to the query sphere(s1,r,2) $\wedge$ cube(c1,g,1) (with s1 and c1 the identifiers of the entities).

The *high-level features* that we employ in the present paper are derived from the E/R diagram. These can be obtained by replacing the values for the attributes in the low-level features by *variables* in the query (represented by a capital letter or capitalized string, e.g. Col). At present we consider single vertices or paths of length l starting from nodes representing entities. For instance, selecting the vertex for sphere would result in the query sphere(S,Col,W). This corresponds to all sphere entities in the unrolled E/R diagrams of the example instances. The path of length 3 sphere - next_sc - cube corresponds to the query sphere(S,Col1,W1) $\wedge$ next_sc(S,C) $\wedge$ cube(C,Col2,W2). This selects all sphere and cube entities that are linked by a next_sc relation in the unrolled E/R diagrams of the instances. The difference between the low and high-level features is thus

[1] More specifically, as explained in Section 3, kLog is based on the NSPDK kernel [16], so we shall consider all pairs of near neighborhood subgraphs.

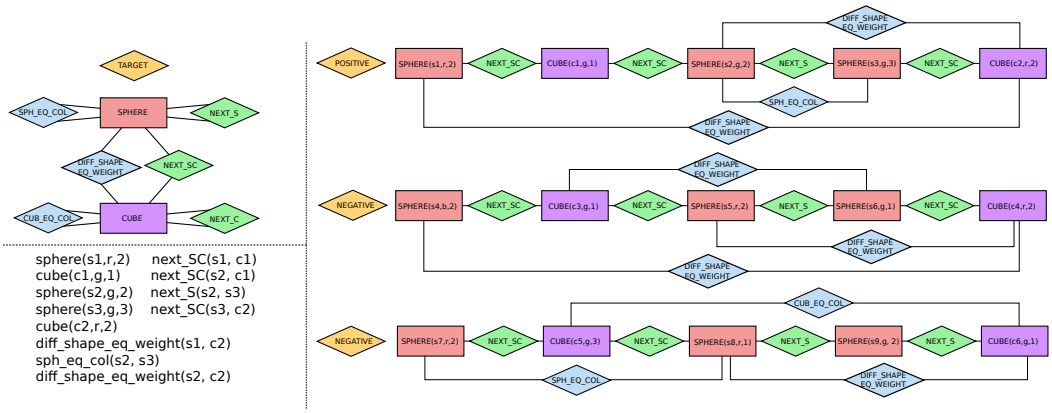

Figure 1: Introductory example – Top left: E/R diagram modeling the domain. Bottom left: example interpretation for the positive example (top right). Right: graphicalizations (i.e., unrolled E/R) for 3 instances.

that the high-level features do not contain any constants and are represented by a subgraph of the E/R-diagram, whereas the low-level ones contain values for all the involved attributes, and are represented by subgraphs of the unrolled E/R-diagram.

The definition of the high-level features (through the E/R diagram) is again reminiscent of inductive logic programming systems and constitutes the declarative bias of the system; the reader should keep in mind that alternative biases would be easy to incorporate.

Notice that one can look for occurrences of a high-level feature in the database. This can be realized either at the relational level (using, e.g., Prolog's query answering mechanism) or by graph matching (subgraph isomorphism testing between the feature and the graphicalized database).

In order to tie low-level features to high-level ones, we consider where the high-level feature occurs in the graphicalized examples. All the low-level graph features that are in the neighborhood of such an occurrence are collected in what we call the *coverage* of the high-level feature (see Figure 3). The underlying assumption is that predicates, that are near in the graphicalized representation (in terms of shortest path distance), are likely to be more related in some meaningful way. Both the *relational regularization* and the *relational feature selection* techniques are based on this notion: the *regularization* is achieved tying the importance score of all low-level features that belong to the same coverage; while *feature selection* is obtained by ranking the high-level features according to the aggregated score of all low-level features that belong to the respective coverage. These notions are formally elaborated in Sections 4 and 5.

Experimental Results To illustrate our regularization setting upfront, we already provide the results on our toy example. As candidate high-level relational features we consider the single vertices in the E/R diagram together with paths of the type entity-relation-entity. The resulting *relational feature ranking* is as follows: sphere - sph_eq_col - sphere= 22.90, sphere=2.11, cube= -1.48, sphere - diff_shape_eq_weight - cube= -3.62, cube - diff_shape_eq_weight - sphere= -3.62 and cube - cub_eq_col - cub=-11.78. As expected, the sph_eq_col relation between two spheres is ranked highest, followed by spheres and cubes. The relations between two cubes are ranked lowest (i.e., they are highly discriminative for the negative examples). diff_shape_eq_weight relations between two distinct objects can be seen as neutral high-level features.

We also compared the performance of relational regularization to a standard regularization on a simple propositionalized representation. This latter case is obtained by considering a bag-of-words representation of the entity and relationship nodes. Both in terms of F-measure and area under the ROC curve, relational regularization shows a significant improvement over the non-relational case, passing from F1 0.82 to 0.89, and from AUC 0.85 to 0.91.

3 From Graphs to Features

The aim of relational feature selection is to identify important high-level features in a relational dataset. In the following sections we make these intuitive notions more precise.

We start by introducing the technique that we use to map the graphicalized instances into sparse feature vectors living in a very high dimensional space. The main difference between this way of extracting features from relational data and other propositionalization techniques lies in its implicit and combinatorial nature: instead of asking the user to explicitly define the predicates that have to be considered as features, here we use an approach derived from a graph kernel technique. Informally, we extract as features all possi-

ble pairs of small subgraphs that are near each other. This approach yields feature spaces with size up to 10^{10}, spaces that are richer than most of the other direct propositionalization approaches[2].

3.1 Graph Kernel Decomposition The features are based on the decomposition [17] performed by the graph kernel, called Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [16]. Informally the idea is to decompose a graph into *"parts"*, i.e., small, near neighborhood subgraphs of increasing radii $r < r_{max}$. Then, all pairs of such subgraphs whose roots are at a distance not greater than $d < d_{max}$ are considered as individual features. At an intuitive level, it decomposes each large neighborhood of size $r_{max} + d_{max}$ into small fragments. It is these fragments that constitute the *low-level features* in our approach. The kernel notion is finally given as the fraction of common fragments between two graphs. For the sake of self-containment we briefly report the formal definitions.

For a given graph $G = (V, E)$, and an integer $r \geq 0$, let $N_r^v(G)$ denote the subgraph of G rooted in v and induced[3] by the set of vertices $V_r^v \doteq \{x \in V : d(x, v) \leq r\}$, where $d(x, v)$ is the shortest-path distance between x and v. A neighborhood $N_r^v(G)$ is therefore a topological *ball* with center v and radius r (see Figure 2).

We introduce the relation, which is defined in terms of neighborhood subgraphs, as: $R_{r,d} = \{(N_r^v(G), N_r^u(G), G) : d(u, v) = d\}$ that is, a relation $R_{r,d}$ that identifies pairs of neighborhoods of radius r whose roots are exactly at distance d. Subsequently:

$$\kappa_{r,d}(G, G') = \sum_{\substack{A,\, B \in R_{r,d}^{-1}(G) \\ A',\, B' \in R_{r,d}^{-1}(G')}} \mathbf{1}_{A \cong A'} \cdot \mathbf{1}_{B \cong B'} \tag{3.1}$$

where $R_{r,d}^{-1}(G)$ indicates the multiset of all pairs of neighborhoods of radius r with roots at distance d that exist in G, and where $\mathbf{1}$ denotes the indicator function and $\cong$ the isomorphism between graphs. The normalized version of $\kappa_{r,d}$ is $\hat{\kappa}_{r,d}(G, G') = \frac{\kappa_{r,d}(G,G')}{\sqrt{\kappa_{r,d}(G,G)\kappa_{r,d}(G',G')}}$. To increase efficiency and generalization power, the *zero-extension* of K is considered, obtained by imposing upper bounds on the radius and the distance parameter, with values to be determined e.g. via cross validation: $K_{r_{max},d_{max}}(G, G') = \sum_{r=0}^{r_{max}} \sum_{d=0}^{d_{max}} \hat{\kappa}_{r,d}(G, G')$.

Explicit Features While a kernelized learning machine based on NSPDK would only need the efficient computation of $\kappa_{r,d}$, here we want to explicitly extract the feature encoding induced by the graph kernel. This gives us a flexibility in manipulating and combining the importance score associated with each feature, that is not achievable when working in the implicit form.

The technique, based on ideas presented in [16], works in two steps: 1) a graph invariant encoding $\mathcal{L}^g(A)$ is constructed as a sorted list of annotated edges for a neighborhood graph, where the annotation uses an efficient quasi-canonical vertex relabeling; 2) the list is then hashed $H(\mathcal{L}^g(A)) \to \mathbb{N}$ to obtain an integer that is used to compose the final feature identifier.

4 Relational Regularization

We have described how, after the graphicalization step, we extract an explicit low-level graph feature representation for each instance. In the following we first introduce the technique used to link the low-level graph feature representation in the graph to the high-level relational feature representation in the E/R diagram. After that we will introduce the relational regularization scheme used to enforce the dependency of the importance score between all low-level graph features related to the same high-level relational feature.

Definitions: Topological Co-occurence, Co-occurrence Matrix, Coverage, and High-level Relational Features We say that two low-level graph features are *topologically co-occurring* when they are both occurring in the same neighborhood subgraph *and* share at least one root vertex (see Figure 3). Formally, two features $(A, B), (A', B') \in R^{-1}(G)$ co-occur in a local neighborhood if they are of the form $(A, B) = (N_r^v(G), N_r^u(G) : d(v, u) = d)$ and $(A', B') = (N_{r'}^v(G), N_{r'}^{u'}(G) : d(v, u') = d')$. Note that the features need not have the same radius r or the same distance d between roots. Intuitively, d_{max} controls the degree of locality, while r_{max} controls the complexity of the low-level graph features.

Given a dataset of instances $\hat{R}$ and its correspondent set of graphicalized instances $\hat{G}$, we define the *co-occurence matrix* as the matrix W which stores in W_{ij} the aggregate number of times that feature i was topologically co-occurring with feature j in all the graphs in $\hat{G}$. We indicate with $\Phi_{r_{max},d_{max}}^v(G)$ the multiset of all graph features (with radius bounded by r_{max} and distance bounded by d_{max}) that are topologically co-occurring in the neighborhood of a root vertex v with

[2] Note that often a graph kernel is just an efficient procedure to compute a dot product between two instances and the feature representation is only available in an implicit fashion. For those kernels the explicit representation would lead to unacceptable storage and running times. Due to the specific assumptions in our graph kernel however, while maintaining a very large overall feature space dimension, we operate with a small number of features for each instance, with the size being typically linear (with a small multiplicative constant) w.r.t. the input graph size (see [16] for a detailed analysis).

[3] In a graph G, the *induced-subgraph* on a set of vertices $W = \{w_1, \ldots, w_k\}$ is a graph that has W as vertex set and contains every edge of G whose endpoints are in W.

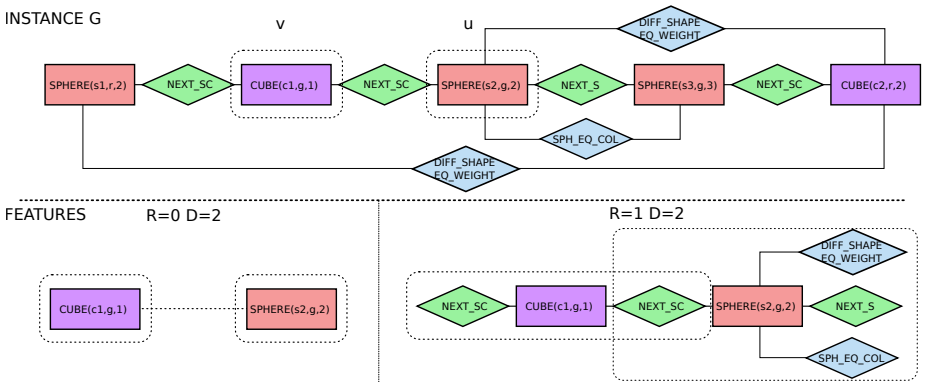

Figure 2: NSPDK features. Top: instance with two vertices v, u selected as roots for neighborhood subgraphs features at distance 2. Bottom: neighborhood pairs (NSPDK features) at distance 2 with radius R=0 and 1 respectively. Note that neighborhood subgraphs can overlap.

radius d_{max} and that have v as common root vertex.

Let $H \in \mathcal{H}$ be the (graph) representation of an E/R model; a *high-level relational feature* then corresponds to a (homomorphic) subgraph h of H. In the simplest case h can be a single vertex in H. We now define a mapping $\Pi : \mathcal{H} \times \mathcal{R} \mapsto \mathcal{G}$ which, for a given example instance $R \in \mathcal{R}$ (for which there exist a graphicalization $g(R) = G \in \mathcal{G}$) maps a subgraph h of the E/R diagram H to the corresponding subgraphs[4] of G in the graphicalized domain, that is, we map high-level features to low-level subgraphs. Let us now consider the set of all vertices in all graphs that are the result of the mapping of a given high-level feature. We are interested in the low-level features that can be generated starting from these vertices: we call this multiset the *coverage* of the high-level feature. Formally, given a subgraph $h \in H$ of an E/R diagram, we define the *coverage of* h as the multiset:

$$C(h) = \bigcup_{R \in \hat{R}} \bigcup_{v \in V(\Pi(h,R))} \Phi^{v}_{r_{max},d_{max}}(g(R))$$

where R is taken over all instances in a dataset $\hat{R}$ and v is taken in all vertices that map to h according to the unfolding of the E/R element in each graphicalized instance $g(R) = G$. Using the coverage notion we can link the importance of high-level relational features with the importance of low-level graph features: given an importance score vector θ associated to the low-level graph features, we define the importance score S of a subgraph h in E/R as the sum of the scores for the features in its coverage: $S(h) = \sum_{\phi_i \in C(I)} \theta_{\phi_i}$.

In the present implementation we do not consider sophisticated ways to construct the high-level feature domain, but instead consider only single vertices or paths of length 3, that is, triplets of the form entity-relation-entity. In future work we will consider a combinatorial decomposition approach to produce more complex high-level feature domains.

Relational Regularization with Co-occurence Constraints In the following we introduce the relational regularization scheme used to enforce dependency between all low-level graph features related to the same high-level relational feature.

The regularization technique is best introduced together with an associated learning algorithm. Here we employ models of the linear type, i.e., $f(x) = \theta^\top x$, where $x \in \mathbb{R}^p$ is a vector input in the high-dimensional space induced by the graph decomposition approach detailed in Section 3.1 and $\theta \in \mathbb{R}^p$ is the associated parameter vector. Given a dataset of input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ describing a learning problem, and a choice for a loss function $\ell(\hat{y}, y)$, we look for a parameter assignment that minimizes the empirical risk $E_n(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)$. We solve the minimization problem with a stochastic gradient descent technique, which simplifies the ordinary gradient descent algorithm and estimates the gradient direction on a per instance basis with the following iterative update[5]: $\theta_{t+1} = \theta_t - \gamma \nabla_\theta \ell(f(x_i), y_i)$. As loss function we consider the regularized SVM loss: $\ell(f(x_i), y_i) = max\{0, 1 - y_i \theta^\top x_i\} + \lambda \theta^\top \theta$.

We now introduce the *relational regularizer*. Given the co-occurence matrix W we build the Laplacian matrix $L = D - W$, with diagonal matrix D having entries $D_{ii} = \sum_j W_{ij}$, and its normalized version $\hat{L} = D^{-1/2} L D^{-1/2}$. Imposing a penalty on the size of $\theta^\top \hat{L} \theta$ is equivalent to penalize the difference between parameter θ_i and parameter θ_j when W_{ij} is large. This can be seen considering that $\theta^\top L \theta = \frac{1}{2}\sum_{i,j}(\theta_i - \theta_j)^2 W_{ij}$, hence for a given large W_{ij}, minimizing $\theta^\top L \theta$

[4] Note that these subgraphs need not be connected.

[5] We employ a decreasing gain $\gamma \approx \frac{1}{t}$ to ensure fast convergence as suggested by [18].

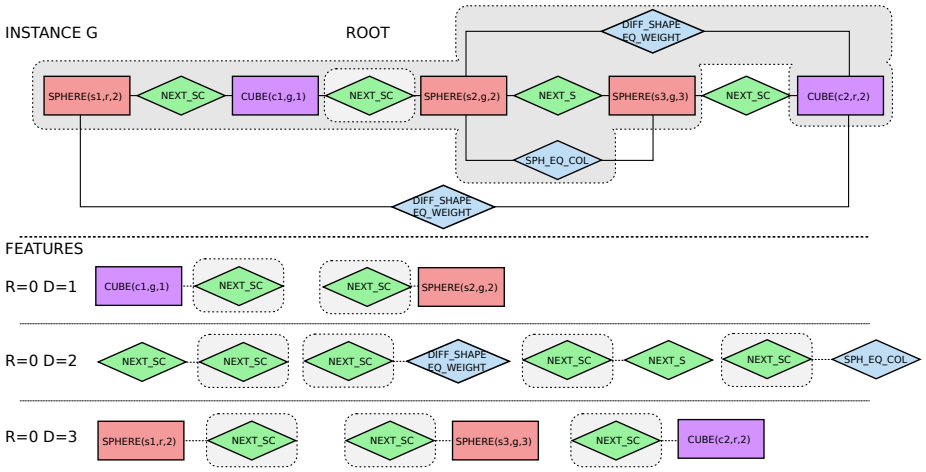

Figure 3: Feature co-occurrence. Top: neighborhood with $d_{max} = 3$ for root node of type NEXT_SC. Bottom: co-occurring features with $r_{max} = 0$: (R=0, D=1), (R=0, D=2) and (R=0, D=3).

implies enforcing $\theta_i - \theta_j \rightarrow 0$. To see why, we rewrite $\sum_{i,j}(\theta_i - \theta_j)^2 W_{ij} = \sum_i \theta_i^2 D_{ii} + \sum_j \theta_j^2 D_{jj} - 2\sum_{i,j}\theta_i\theta_j W_{ij} = 2\theta^\top L\theta$. Similar considerations hold for $\hat{L}$. In practice this implies that the regularization term $\theta^\top \hat{L}\theta$ enforces the binding of parameters associated to features that are topologically co-occurring.

Adding this novel regularization term to the loss function we obtain:

$$\ell(f(x_i), y_i) = max\{0, 1 - y_i\theta^\top x_i\} + \lambda\theta^\top\theta + \rho\theta^\top\hat{L}\theta.$$

Taking the derivatives gives us the corresponding new update rule for the stochastic gradient descent algorithm:

$$\begin{aligned} &\theta_{t+1} = \theta_t - \gamma_t(\lambda\theta + \rho\hat{L}\theta) && \text{if } y_t\theta^\top x_t > 1 \text{ and} \\ &\theta_{t+1} = \theta_t - \gamma_t(-y_t x_t + \lambda\theta + \rho\hat{L}\theta) && \text{otherwise} \end{aligned}$$

We keep the regularization term $\theta^\top\theta$ in place since, at times, the relational manifold assumption can hold to a lesser degree. In this case it is important to be able to trade off between the two regularizers via λ and ρ.

Note that in principle, our notion of locality and regularization could be applied to any dataset that can be interpreted as an E/R diagram with corresponding features. For example, in the context of Markov Logic our notion of locality might be applied to the grounded Markov Logic network where the (ground) clauses would act as the features and the entities would be the ground atoms.

5 Relational Feature Selection

There are three main strategies to perform feature subset selection [19], namely filter (a pre-processing step independent from any downstream learner), embedded (in which learning is interleaved with the feature selection procedure) and wrapper (which uses the learner as a black-box). Here we present an embedded and a wrapper strategy, both employing a linear model equipped with the relational smoother introduced in Section 4. The intuitive difference between the two approaches is that while the (more computationally efficient) embedded method is forced to consider the full E/R model all at once, the wrapper method analyzes the importance of isolated subsets of interacting entities and relations, modifying the E/R model under examination.

5.1 Embedded Strategy Embedded methods incorporate variable selection as part of the training phase and are tightly linked to the learning algorithm.

Our approach is rooted in the AROM (Approximation of the zeRO norm Minimization) method [20]. It is an approximation for the following optimization problem: $\min_\theta ||\theta||_0^0$ subject to: $y_i(\theta^\top x_i) \geq 1$. Here the solution is a separating hyperplane with the minimal number of nonzero elements in the parameter vector θ. Since this is a combinatorially hard problem [21], Weston et al. [20] suggest to minimize instead: $\min_\theta \sum_{j=1}^p \ln(\epsilon + |\theta_j|)$ subject to: $y_i(\theta^\top x_i) \geq 1$ for which they can prove the bound: $||\theta_l||_0^0 \leq ||\theta_0||_0^0 + O\left(\frac{1}{\ln\epsilon}\right)$ where θ_l is the minimizer of the proposed approximation and θ_0 is the minimizer of the original problem. The idea is that, because of the form of the logarithm (i.e., it decreases quickly to zero compared to how fast it increases for large values), significantly increasing a specific θ_j can be compensated by setting to zero another θ_i rather than attempting some compromise between them, thus encouraging sparse solutions. A similar reasoning holds in the case of non (linearly) separable data.

Interestingly, they proposed to solve the optimization problem with a very simple algorithm (see Algorithm 1), where input vectors are iteratively scaled by the optimization solution θ. Here, differently from We-

Algorithm 1 Embedded Relational Feature Selection

Input: predicate set S, radius r, distance d,
Output: importance for each element in set S
Note: {"$\cdot$" is a component-wise multiplication}
G = RelationalDBGraphicalization(S)
X = ExplicitFeatureExtraction(G, r, d) {$\triangleright$ X is the set of all vector representations}
$z \leftarrow (1, \ldots, 1)$ {$\triangleright$ z is a vector of scaling factors}
repeat
  for $x_i \in X$ **do**
    $\bar{x}_i \leftarrow x_i \cdot z$ {$\triangleright$ rescale all instances}
  end for
  let θ^* be the solution of the SVM problem on $\bar{X}$ with relational regularization
  $z \leftarrow z \cdot \theta^*$ {$\triangleright$ update scaling factors}
until convergence
$\hat{z} \leftarrow$ binarize(z) {$\triangleright$ set to 1 nonzero features}
for $x_i \in X$ **do**
  $\hat{x}_i \leftarrow x_i \cdot \hat{z}$ {$\triangleright$ consider only nonzero features for each x_i}
end for
let $\hat{\theta}^*$ be the solution of the SVM problem on $\hat{X}$ with relational regularization
return ComputePredicateImportance($\hat{\theta}^*$)

ston et al. [20], we solve a SVM optimization problem with the additional constraint of the relational regularizer. As suggested in [20] in the last step we binarize the scaling vector z to have entries in $\{0, 1\}$. Only at this point we solve the SVM problem on the selected subset of nonzero features. This is an important step, given that the iterative algorithm scales θ as a function of the iteratively *rescaled* input x_i and, in order to be used on the original input, it would require a compensatory rescaling.

5.2 Wrapper Strategy Our wrapper method is a derivation of the Random Sets approach [22]. The method receives as input the set of the predicates to be ranked and the number of random subsets of predicates to try out. The idea is to induce a predictive model on each random set of predicates, retain only the top performing models, and select the high-level relational features that are ranked highest among these models.

When working in a propositional setting, there are generally no constraints among the features and one can sample sets of features uniformly at random. In a *relational* setting, however, given the logical dependencies between predicates, there are consistency issues. As an example, in the definition of our domain knowledge we could have predicate A defined in terms of predicate B; in this case removing B would necessarily imply the removal of A. To deal with this issue we make use of a dependency graph, built automatically from the background knowledge, to guarantee the generation of consistent E/R models.

A predictive model equipped with the relational regularizer is learned on the graphs resulting from applying the graphicalization procedure to the reduced E/R model. A user specified number of the best predictive linear models are selected. Finally the importance score of each high-level feature is computed based on the sum of the weights for the low-level graph features corresponding to each high-level feature's coverage.

6 Experiments

We validated our techniques on two tasks with structural input, where relational learning, and kLog in particular, has proven to be a promising approach. The first task, *hedge cue detection*, is a natural language processing problem. The second, *molecular toxicity prediction*, is a classification task in bioinformatics.

6.1 Natural Language Processing: Hedge Cue Detection Hedge cue detection is a binary classification problem in which the task is to determine whether a sentence contains hedge cues. Hedge cues are linguistic devices that indicate whether information is being presented as uncertain or unreliable within a text [23]. Since this task involves processing non-propositional aspects of meaning, the traditional local token-based approaches based on the lexico-syntactic features of individual words no longer suffice. As shown by [24], adding relational information and solving the problem with SRL can improve performance.

In this task, the instances are sentences and consist of a number of words w, for which the order is represented by the *next* relation. The dependency relations represent the structure of syntactic relations between the words. This is modeled by $depHead$, where $depRel$ specifies the type of the dependency (e.g., subject, object, etc.). Other properties of the word that are taken into account as features are the word string itself, its lemma, the Part-of-Speech tag, the chunk tag and a binary feature that represents whether the word is part of a predefined list of speculative strings. $weaselSentence$ represents the target relation.

As in Verbeke et al. [24], we also used three additional predicates to encode additional background knowledge; CW retains only the words that appear in a predefined list of weasel words compiled from the training data, whereas LeftOf and RightOf represent two surrounding words in the sentence with their respective lemmas and PoS tags. Since the latter predicates have shown to improve performance in previous results for this task, it is to be expected that these are the main discriminative predicates, while the propositional lexico-syntactic features should be less informative.

In a first step, the high-level feature importance was calculated by both the embedded and the wrapper approach, for which the results are listed in Table 1.

Embedded approach			Wrapper approach		
#	Feature	Score	#	Feature	Score
1	CW	27.20	1	RightOf	15.99
2	RightOf	6.69	2	CW	13.72
3	LeftOf	4.30	3	LeftOf	9.68
4	Next	4.02	4	Next	5.81
5	DH	3.42	4	DH	2.97
5	WordString	-2.35	5	WordString	-2.06
5	InList	-2.35	6	InList	-2.56
5	Chunk	-2.35	7	Chunk	-2.71
5	Lemma	-2.35	8	Lemma	-2.72
5	PoS	-2.35	9	PoS	-2.90

Table 1: High-level feature ranking (Hedge cue detection)

Triplet	Score
cw - next - cw	49.79
cw - dh - cw	49.73
cw - dh - word	37.29
cw - next - word	31.79
word - dh - word	12.62
word - next - word	-5.76

Table 2: Triplet ranking (Hedge cue detection)

As can be seen, both approaches give a similar ranking, except for RightOf and CW, which are switched, and the basic lexico-syntactic features, which get an equal weight in the embedded method, and slightly different scores in the wrapper method.

Table 2 contains the ranking of the possible triplets of predicates of type entity - relationship - entity. It can be observed that pairs of consecutive hedged words or hedged words that are linked in the dependency tree are very informative for the task at hand.

As for the artificial example, we compared the performance of relational regularization to standard regularization on a propositionalized representation, obtained by considering a bag-of-words representation of the entities and relations nodes. As a baseline, we compared to the performance of the classification without regularization, again both in the relational and the propositional case. The results can be found in Table 3. In the case of hedge cue detection, it can be observed that the relational regularization shows a significant improvement over the non-relational case. When comparing to the respective results when no regularization is used, we observe similar results[6]. This indicates that the right features are selected during the regularization process, as learning with a reduced number of features does not come at the cost of a reduced performance.

	Relational regularization	Non-rel. regularization	Relational baseline	Non-rel. baseline
Hedge Cue Detection				
Precision	59.98	33.19	60.61	33.19
Recall	50.76	96.42	49.28	96.42
F1	54.99	49.38	54.36	49.38
Molecular Toxicity				
Precision	79.69	55.81	82.37	57.14
Recall	74.13	54.21	70.02	44.35
F1	76.81	55.00	75.69	49.94

Table 3: Relational vs. non-relational regularization

6.2 Chemoinformatics: Molecular Toxicity In Kazius et al. [25] a dataset of 4337 molecular structures is constructed. The dataset is available with corresponding Ames target data, i.e., each molecule has been tested in a short-term in vitro assay designed to detect genetic damage caused by chemicals and has become the standard test to determine mutagenicity. The natural relational representation in terms of atoms and bonds is enriched with a chemistry knowledge base offered by DMax Chemistry Assistant, which allows the determination of the aromaticity perception, the annotation of functional groups, and the relations between them.

The dataset was split in a train and test partition of 3337 versus 1000 molecular structures respectively. In Figure 4, the performance in terms of ROC, F1 and accuracy is plotted against the number of iterations, along with the number of retained features (right y axis, in logarithmic scale). We report a substantial stability of the predictive performance w.r.t. the number of features, an indication that relational regularization is effective in selecting the important high-level features.

The analysis of the high-level relational feature ranking (Table 4) reveals that the toxicity concept is related to the way functional groups and atoms are connected (fg_connected and bond) rather than on the actual type of atoms and type of functional group (fg_member and atom). It is in fact known that one component in the mutagenicity capacity of a substance depends on the presence of aromatic rings (bond type) forming rigid planes (connected and fused) that can "cut" into DNA/RNA filaments and break them. The results of the performance comparison of relational versus non-relational regularization can again be found in Table 3. Similar observations as in the case of hedge cue detection can be made.

7 Conclusions

In this paper we have lifted regularization and feature selection to a *relational* level in order to increase performances and achieve higher interpretability. These techniques take into account the relational structure and topology of a domain and are based on a notion

[6]The difference in results for the relational baseline as compared to [24] is due to a smaller grid search parameter range during optimization.

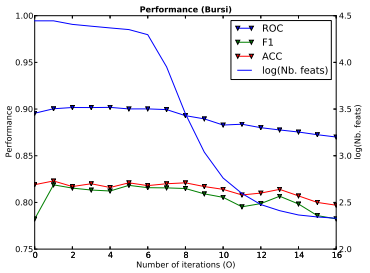

Figure 4: Performance with decreasing number of features.

High-level feat.	Score
fg_connected	33.29
bond	22.42
fgroup	21.96
fg_fused	6.98
atom	-4.44
fg_member	-21.69
fg_linked	-22.78

Table 4: High-level feature ranking (Mol. tox.)

Triplet	Score
fgroup - fg_connected - fgroup	63.86
fgroup - fg_fused - fgroup	38.72
fgroup - fg_linked - fgroup	28.32
atm - bnd - atm	17.31
fg_member - atm - fgroup	-4.84

Table 5: Triplet ranking (Molecular toxicity)

of locality that ties together relevant features in the entity-relationship model. Finally, we have presented two *relational feature selection* approaches; a wrapper technique, which manipulates the declarations in the language bias used by the learner; and an embedded method that incorporates the feature selection as part of the training phase.

Acknowledgements

This research is funded by the Research Foundation Flanders (FWO project G.0478.10) and by the German Research Foundation (DFG-grant BA 2168/3-1). The authors would like to thank Kurt De Grave for his valuable feedback.

References

[1] Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679

[2] Getoor, L., Taskar, B., eds.: An Introduction to Statistical Relational Learning. MIT Press (2007)

[3] De Raedt, L.: Logical and Relational Learning. Springer (2008)

[4] Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining. In: Relational Data Mining. Springer (2001) 262–291

[5] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (February 2006) 107–136

[6] Krogel, M.A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In Rouveirol, C., Sebag, M., eds.: Proc. of the 11th Int. Conf. on Inductive Logic Programming. Volume 2157 of LNAI., Springer (2001) 142–155

[7] Frasconi, P., Costa, F., De Raedt, L., De Grave, K.: klog: A language for logical and relational learning with kernels. CoRR **abs/1205.3981** (2012)

[8] Srinivasan, A.: The Aleph Manual. (2007)

[9] Blockeel, H., De Raedt, L.: Top-down induction of first order logical decision trees. Artificial Intelligence **101**(1-2) (June 1998) 285–297

[10] Alphonse, E., Matwin, S.: Feature subset selection and inductive logic programming. In: Proc. Int. Conf. on Machine learning. (2002) 11–18

[11] Jensen, D., Neville, J.: Linkage and autocorrelation cause feature selection bias in relational learning. In: Proc. Int. Conf. on Machine learning. (2002) 259–266

[12] Neville, J., Jensen, D.: Leveraging relational autocorrelation with latent group models. In: Proc. of the 5th IEEE Int. Conf. on Data Mining. (2005) 322–329

[13] Huynh, T., Mooney, R.: Discriminative structure and parameter learning for markov logic networks. In: Proc. Int. Conf. on Machine learning. (2008) 416–423

[14] Quanz, B., Huan, J.: Aligned graph classification with regularized logistic regression. In: SDM. (2009) 353–364

[15] Zhou, D., Schölkopf, B.: A regularization framework for learning from graph data. In: Proc. of the ICML Workshop on Statistical Relational learning. (2004)

[16] Costa, F., De Grave, K.: Fast neighborhood subgraph pairwise distance kernel. In: Proc. of the Int. Conf. on Machine Learning. (2010) 255–262

[17] Haussler, D.: Convolution kernels on discrete structures. Technical Report 99-10, UCSC-CRL (1999)

[18] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer (2010) 177–186

[19] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182

[20] Weston, J., Elisseeff, A., Schölkopf, B., Tipping, M.: Use of the zero norm with linear models and kernel methods. JMLR **3** (2003) 1439–1461

[21] Amaldi, E., Kann, V.: On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theoretical Computer Science **209**(1) (1998) 237–260

[22] Nikulin, V.: Random sets approach and its applications. Causation and Prediction Challenge Challenges in Machine Learning **2** (2008) 59

[23] Hyland, K.: Hedging in scientific research articles. Pragmatics & Beyond: New Series. J. Benjamins Publishing Company (1998)

[24] Verbeke, M., Frasconi, P., Van Asch, V., Morante, R., Daelemans, W., De Raedt, L.: Kernel-based logical and relational learning with kLog for hedge cue detection. In: Proc. of the Int. Conf. on Inductive Logic Programming. (2012) 347–357

[25] Kazius, J., McGuire, R., Bursi, R.: Derivation and validation of toxicophores for mutagenicity prediction. J. Med. Chem. **48**(1) (2005) 312–320